In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
```

Loading data from csv

In [2]:

```
path = "../input/bbc/bbc-text.csv"
```

In [3]:

```
data = pd.read_csv(path)
data_size = len(data)
```

In [4]:

```
data.head()
```

Out[4]:

	category	text
0	tech	tv future in the hands of viewers with home th...
1	business	worldcom boss left books alone former worldc...
2	sport	tigers wary of farrell gamble leicester say ...
3	sport	yeading face newcastle in fa cup premiership s...
4	entertainment	ocean s twelve raids box office ocean s twelve...

In [5]:

```
data.tail()
```

Out[5]:

	category	text
2220	business	cars pull down us retail figures us retail sal...
2221	politics	kilroy unveils immigration policy ex-chatshow ...
2222	entertainment	rem announce new glasgow concert us band rem h...
2223	politics	how political squabbles snowball it s become c...
2224	sport	souness delight at euro progress boss graeme s...

Calculating percentage of each class

In [6]:

```
# category percent among the total data
data['category'].value_counts()/ data_size * 100
```

Out[6]:

```
sport            22.966292
business         22.921348
politics         18.741573
tech             18.022472
entertainment    17.348315
```

```
entertainment    17.348315
Name: category, dtype: float64
```

View text data

In [7]:

```
text_data = data[:]
print("data count :-",len(text_data))
```

```
data count :- 2225
```

In [8]:

```
text_data[:5]
```

Out[8]:

	category	text
0	tech	tv future in the hands of viewers with home th...
1	business	worldcom boss left books alone former worldc...
2	sport	tigers wary of farrell gamble leicester say ...
3	sport	yeading face newcastle in fa cup premiership s...
4	entertainment	ocean s twelve raids box office ocean s twelve...

In [9]:

```
duplicate_text_data = text_data[text_data.duplicated()]
print("duplicate data count :-",len(duplicate_text_data))
```

```
duplicate data count :- 99
```

In [10]:

```
duplicate_text_data
```

Out[10]:

	category	text
85	politics	hague given up his pm ambition former conser...
301	politics	fox attacks blair s tory lies tony blair lie...
496	tech	microsoft gets the blogging bug software giant...
543	business	economy strong in election year uk businesse...
582	entertainment	ray dvd beats box office takings oscar-nominat...
...	...	...
2206	politics	kennedy questions trust of blair lib dem leade...
2207	tech	california sets fines for spyware the makers o...
2213	tech	progress on new internet domains by early 2005...
2215	tech	junk e-mails on relentless rise spam traffic i...
2217	tech	rings of steel combat net attacks gambling is ...

99 rows × 2 columns

In [11]:

```
index_of_duplicate_data = duplicate_text_data.index
```

```
index_of_duplicate_data[:5]
```

Out[11]:

```
Int64Index([85, 301, 496, 543, 582], dtype='int64')
```

In [12]:

```
unique_data = data.drop(index_of_duplicate_data)
print("unique data count :-", len(unique_data))
```

```
unique data count :- 2126
```

In [13]:

```
# category percent among the unique data
print("Duplicate Article Diff")
(data['category'].value_counts()- unique_data['category'].value_counts())
```

```
Duplicate Article Diff
```

Out[13]:

```
business          7
entertainment    17
politics         14
sport             7
tech             54
Name: category, dtype: int64
```

In [14]:

```
print("------ Origional Data Stats -----\n",data['category'].value_counts() / data_size * 100)

print("\n\n------ Unique Data Stats -----\n",unique_data['category'].value_counts() / data_size * 1
00)
```

```
------ Origional Data Stats -----
 sport            22.966292
business         22.921348
politics         18.741573
tech             18.022472
entertainment    17.348315
Name: category, dtype: float64


------ Unique Data Stats -----
 sport            22.651685
business         22.606742
politics         18.112360
entertainment    16.584270
tech             15.595506
Name: category, dtype: float64
```

Plotting most frequent words in each category

In [15]:

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

In [16]:

```
tech = unique_data[unique_data["category"] == "tech"]
```

In [17]:

```
tech.tail()
```

Out[17]:

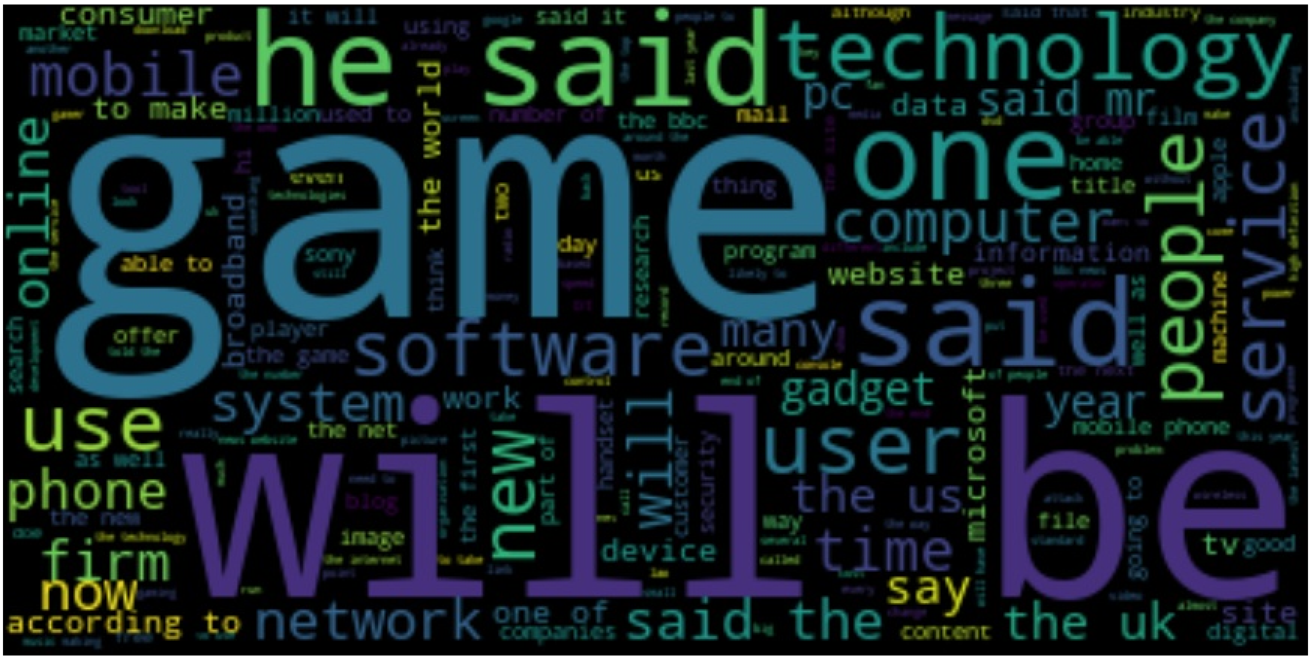
	category	text
2183	tech	piero gives rugby perspective bbc sport unveil...
2189	tech	mobile networks seek turbo boost third-generat...
2200	tech	uk pioneers digital film network the world s f...
2202	tech	local net tv takes off in austria an austrian ...
2204	tech	argonaut founder rebuilds empire jez san the ...

In [18]:

```
techtext = " ".join(tech.text)
```

In [19]:

```
# Create and generate a word cloud image:
wordcloud = WordCloud().generate(techtext)

# Display the generated image:
plt.figure(figsize=(100,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



In [20]:

```
sport = unique_data[unique_data["category"] == "sport"]
```

In [21]:

```
sport.tail()
```

Out[21]:

	category	text
2190	sport	newry to fight cup exit in courts newry city a...
2195	sport	owen delighted with real display michael owen ...
2209	sport	time to get tough on friendlies for an intern...

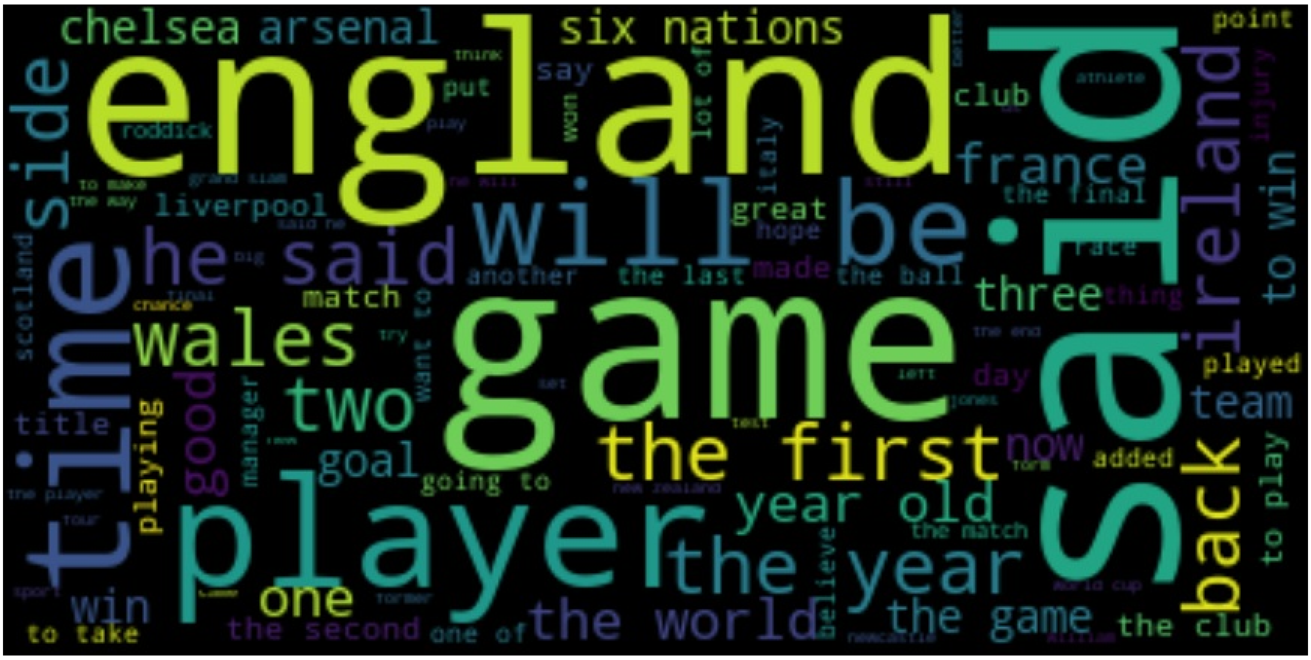
	category	text
2218	sport	davies favours gloucester future wales hooker ...
2224	sport	souness delight at euro progress boss graeme s...

In [22]:

```
sporttext = " ".join(sport.text)
```

In [23]:

```
# Create and generate a word cloud image:
wordcloud = WordCloud().generate(sporttext)

# Display the generated image:
plt.figure(figsize=(100,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



In [24]:

```
business = unique_data[unique_data["category"] == "business"]
```

In [25]:

```
business.tail()
```

Out[25]:

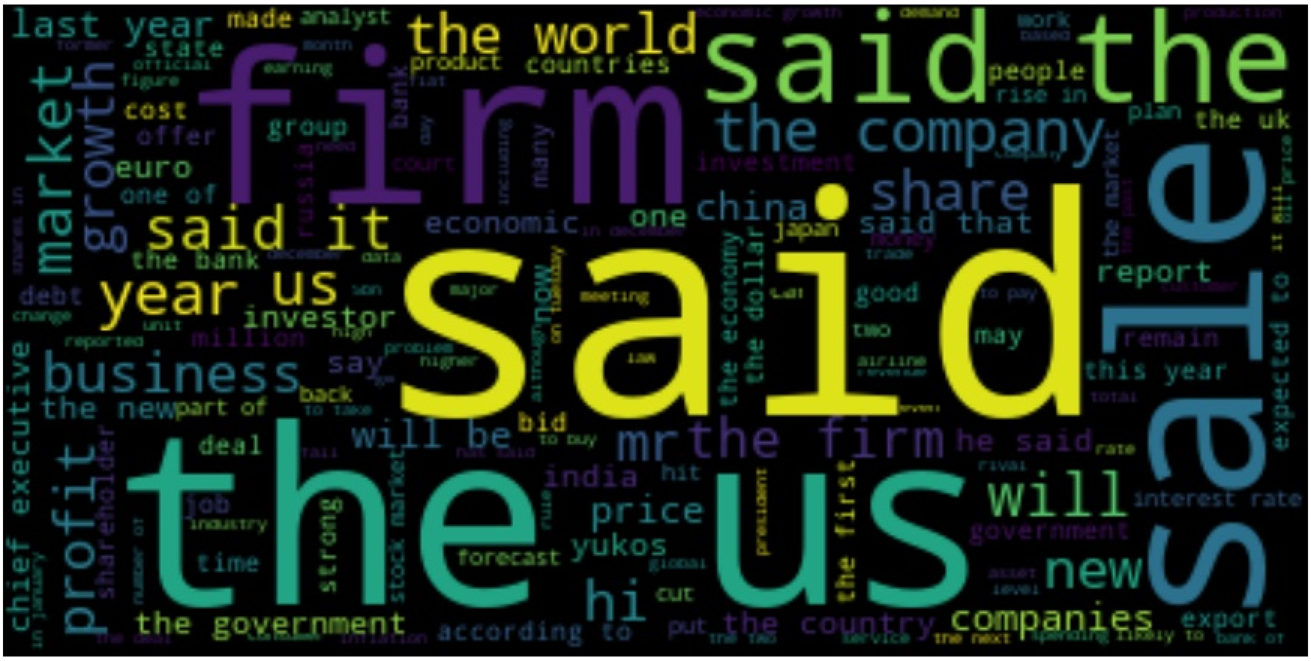
	category	text
2201	business	ban on forced retirement under 65 employers wi...
2212	business	christmas shoppers flock to tills shops all ov...
2214	business	bush budget seeks deep cutbacks president bush...
2219	business	beijingers fume over parking fees choking traf...
2220	business	cars pull down us retail figures us retail sal...

In [26]:

```
businesstext = " ".join(business.text)
```

In [27]:

```
# Create and generate a word cloud image:
wordcloud = WordCloud().generate(businesstext)

# Display the generated image:
plt.figure(figsize=(100,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



In [28]:

```
entertainment = unique_data[unique_data["category"] == "entertainment"]
```

In [29]:

```
entertainment.tail()
```

Out[29]:

	category	text
2205	entertainment	dance music not dead says fatboy dj norman coo...
2208	entertainment	snicket tops us box office chart the film adap...
2211	entertainment	lopez misses uk charity premiere jennifer lope...
2216	entertainment	top stars join us tsunami tv show brad pitt r...
2222	entertainment	rem announce new glasgow concert us band rem h...

In [30]:

```
entertainmenttext = " ".join(entertainment.text)
```

In [31]:

```
# Create and generate a word cloud image:
wordcloud = WordCloud().generate(entertainmenttext)

# Display the generated image:
plt.figure(figsize=(100,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
```

```
plt.axis("off")
#plt.savefig('entertainment_frequent.png')
plt.show()
```

In [32]:

```
politics = unique_data[unique_data["category"] == "politics"]
```

In [33]:

```
politics.tail()
```

Out[33]:

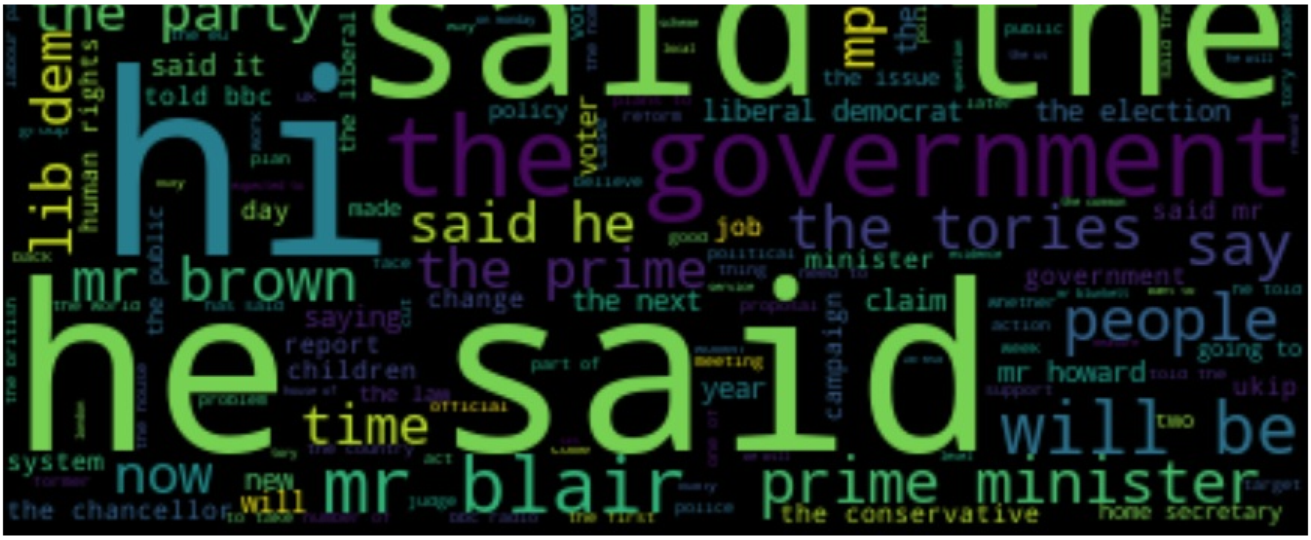
	category	text
2197	politics	campbell returns to election team ex-downing s...
2203	politics	profile: david miliband david miliband s rapid...
2210	politics	teens know little of politics teenagers ques...
2221	politics	kilroy unveils immigration policy ex-chatshow ...
2223	politics	how political squabbles snowball it s become c...

In [34]:

```
politicstext = " ".join(politics.text)
```

In [35]:

```
# Create and generate a word cloud image:
wordcloud = WordCloud().generate(politicstext)

# Display the generated image:
plt.figure(figsize=(100,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
#plt.savefig('politics_frequent.png')
plt.show()
```



Symbols used in articles

In [36]:

```
from collections import Counter

def calculate_frequency(words):
    freqs = Counter(''.join(words.splitlines()))
    for symbol, count in freqs.most_common():
        if not symbol.isalpha() and not symbol.isnumeric():
            print (symbol, count)
```

In [37]:

```
unique_data_text = unique_data.text
len(unique_data_text)
```

Out[37]:

```
2126
```

In [38]:

```
print(unique_data_text[31])
```

```
firefox browser takes on microsoft microsoft s internet explorer has a serious rival in the long-a
waited firefox 1.0 web browser  which has just been released.  few people get excited when some ne
w software is released  especially when the program is not a game or a music or movie player. but
the release of the first full version of firefox has managed to drum up a respectable amount of pr
e-launch fervour. fans of the software have banded together to raise cash to pay for an advert in
the new york times announcing that version 1.0 of the browser is available. the release of firefox
1.0 on 9 november might even cause a few heads to turn at microsoft because the program is
steadily winning people away from the software giant s internet explorer browser.  firefox has bee
n created by the mozilla foundation which was started by former browser maker netscape back in
1998. much of the development work done since then has gone into firefox which made its first appe
arance under this name in february. earlier incarnations  but which had the same core technology
were called phoenix and firebird. since then the software has been gaining praise and converts  no
t least because of the large number of security problems that have come to light in microsoft s in
ternet explorer. rivals to ie got a boost in late june when two us computer security organisations
warned people to avoid the microsoft program to avoid falling victim to a serious vulnerability.
internet monitoring firm websidestory has charted the growing population of people using the
firefox browser and says it is responsible for slowly eroding the stranglehold of ie. before july
this year  according to websidestory  internet explorer was used by about 95% of web surfers. that
figure had remained static for years. in july the ie using population dropped to 94.7% and by the
end of october stood at 92.9%. the mozilla foundation claims that firefox has been downloaded
almost eight million times and has publicly said it would be happy to garner 10% of the windows- u
sing  net-browsing population. firefox is proving popular because  at the moment  it has far fewer
security holes than internet explorer and has some innovations lacking in microsoft s program. for
instance  firefox allows the pages of different websites to be arranged as tabs so users can
switch easily between them. it blocks pop-ups  has a neat way of finding text on a page and lets y
```

```
ou search through the pages you have browsed. one of the most powerful features of firefox is the
many hundreds of extras  or extensions  produced for it. the mozilla foundation is an open source
organisation which means that the creators of the browser are happy for others to play around with
the core code for the program. this has resulted in many different add-ons or extensions for the b
rowser which now include everything from a version of the familiar google toolbar to a homeland se
curity monitor that keep users aware of current threat levels. firefox  which used to be called
firebird and before that phoenix  also has a growing number of vocal net-based fans. a campaign co
-ordinated by the spread firefox website attempted to raise the $50 000 needed for a full page adv
ert in the new york times. the campaign set itself a target of recruiting 2500 volunteers. ten day
s in to the campaign 10 000 people had signed up and now about $250 000 has been raised. the ad is
due to run sometime in a three-week period in late november/early december. the surplus cash will
be used to help keep the mozilla foundation running. microsoft is facing a growing challenge to ie
s hold on the web using population. from alternative browsers such as opera  safari  amaya and eve
n netscape.
```

In [39]:

```
calculate_frequency(unique_data_text[31])
```

```
  653
. 34
- 9
% 4
$ 2
/ 1
```

In [40]:

```
all_text = " ".join(unique_data_text)
len(all_text)
```

Out[40]:

```
4812764
```

Calculating and plotting the total number of words in each doc

In [41]:

```
calculate_frequency(all_text)
```

```
  893007
. 41813
- 12322
) 2126
( 2124
% 1878
: 1652
£ 1335
$ 1187
; 474
& 230
/ 215
! 194
[ 102
] 102
# 28
+ 11
` 3
* 3
= 2
@ 1
```

In [42]:

```
length_count = []
for doc in unique_data_text:
    length_count.append(len(doc))
```

In [43]:

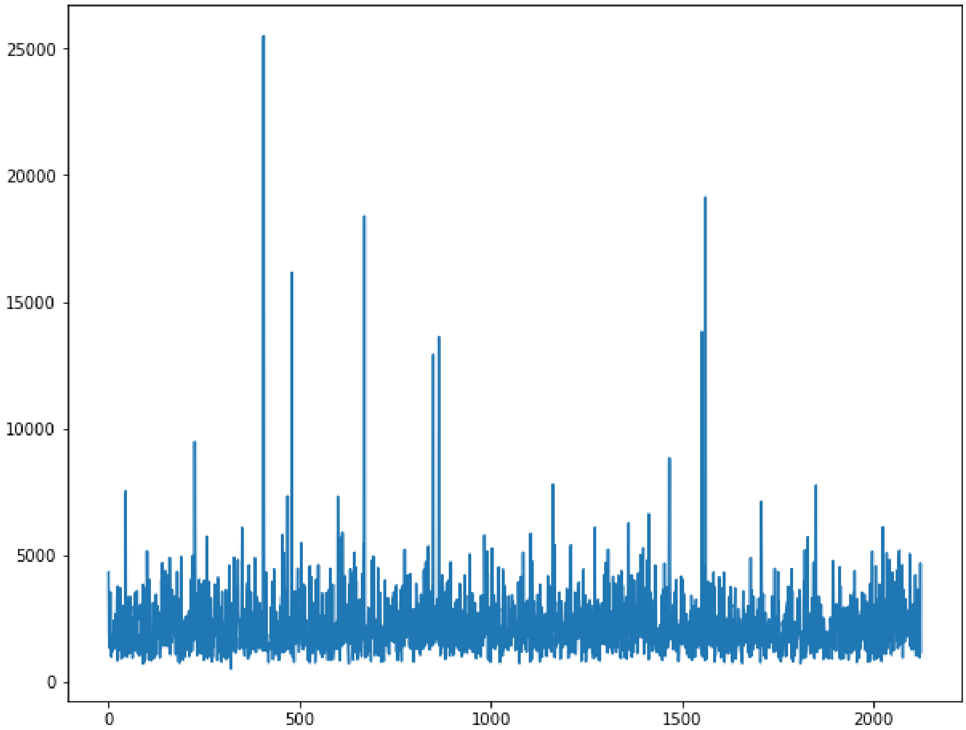
```
from matplotlib import pyplot as plt
```

In [44]:

```
plt.figure(figsize=(10,8))
plt.plot(length_count)
#plt.savefig("sentence_length.png")
```

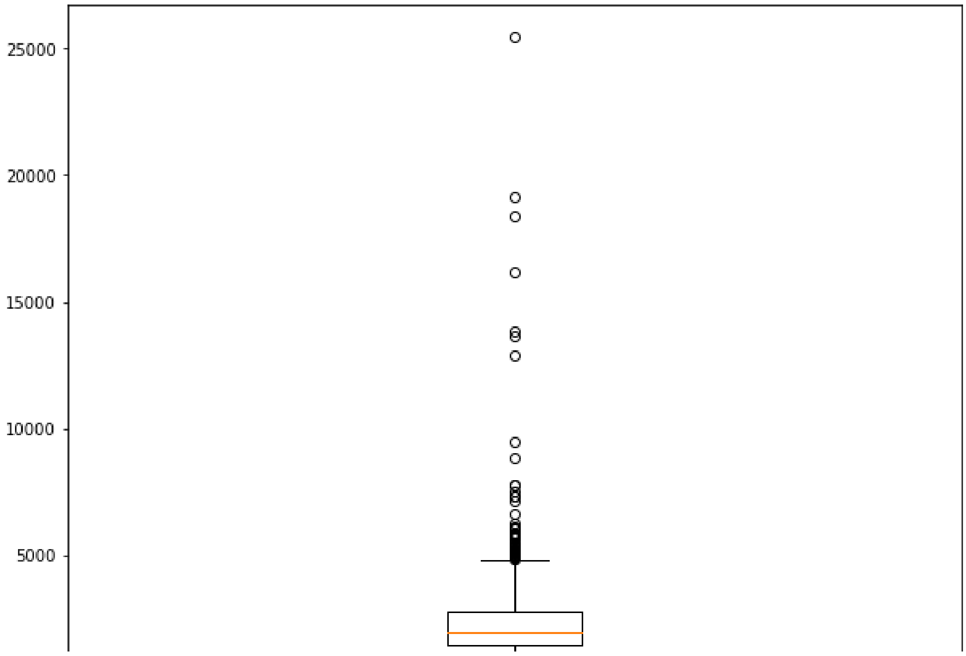
Out[44]:

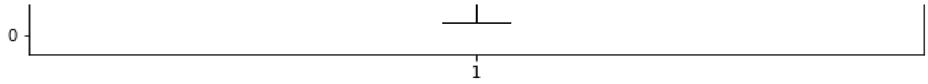
```
[<matplotlib.lines.Line2D at 0x7efe010d0190>]
```



In [45]:

```
plt.figure(figsize=(10,8))
plt.boxplot(length_count)
#plt.savefig("sentence_length_boxplot.png")
plt.show()
```





Total unique words removing stopwords and punctuations

In [46]:

```
import gensim
```

In [47]:

```
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))  # deacc=True removes punc
tuations

data_words = list(sent_to_words(unique_data_text))

# print(data_words[:1])
```

In [48]:

```
len(data_words)
```

Out[48]:

```
2126
```

In [49]:

```
temp = []
for i in data_words:
    temp += i
```

In [50]:

```
len(temp)
```

Out[50]:

```
785607
```

In [51]:

```
unique_words = set(temp)
len(unique_words)
```

Out[51]:

```
27820
```

Joining words as a single entity which occurs more frequently in pairs

Example -> Las_Vegas

In [52]:

```
import numpy as np
```

In [53]:

```
# Build the bigram and trigram models
bigram = gensim.models.Phrases(data_words, min_count=3, threshold=80) # higher threshold fewer phra
ses.
trigram = gensim.models.Phrases(bigram[data_words], threshold=80)

# Faster way to get a sentence clubbed as a trigram/bigram
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)

# See trigram example
```

In [54]:

```
" ".join(trigram_mod[bigram_mod[data_words[0]]])
```

Out[54]:

```
'tv future in the hands of viewers with home theatre systems plasma high_definition_tvs and
digital_video_recorders moving into the living_room the way people watch tv will be radically diff
erent in five years time that is according to an expert panel which gathered at the annual
consumer_electronics_show in las_vegas to discuss how these new technologies will impact one of ou
r favourite pastimes with the us leading the trend programmes and other content will be delivered
to viewers via home networks through cable satellite telecoms companies and broadband
service_providers to front rooms and portable_devices one of the most talked_about technologies of
ces has been digital and personal video_recorders dvr and pvr these set_top_boxes like the us tivo
and the uk sky system allow people to record store play pause and forward wind tv_programmes when
they want essentially the technology allows for much more personalised tv they are also being buil
t in to high_definition tv sets which are big business in japan and the us but slower to take off
in europe because of the lack of high_definition programming not only can people forward wind thro
ugh adverts they can also forget about abiding by network and channel schedules putting together t
heir own la carte entertainment but some us networks and cable and satellite companies are worried
about what it means for them in terms of advertising_revenues as well as brand identity and viewer
loyalty to channels although the us leads in this technology at the moment it is also concern that
is being raised in europe particularly with the growing uptake of services like sky what_happens h
ere today we will see in nine months to years time in the uk adam hume the bbc broadcast
futurologist told the bbc_news_website for the likes of the bbc there are no issues of lost advert
ising_revenue yet it is more pressing issue at the moment for commercial uk broadcasters but brand
loyalty is important for everyone we will be talking more about content brands rather_than network
brands said tim hanlon from brand communications firm starcom mediavest the reality is that with b
roadband_connections anybody can be the producer of content he added the challenge now is that it
is hard to promote programme with so much choice what this means said stacey jolna
senior_vice_president of tv guide tv group is that the way people find the content they want to wa
tch has to be simplified for tv viewers it means that networks in us terms or channels could take
leaf out of google book and be the search_engine of the future instead of the scheduler to help pe
ople find what they want to watch this kind of channel model might work for the younger ipod
generation which is used to taking control of their gadgets and what they play on them but it
might not suit everyone the panel recognised older generations are more comfortable with familiar
schedules and channel brands because they know what they are getting they perhaps do not want so m
uch of the choice put into their hands mr hanlon suggested on the other end you have the kids just
out of diapers who are pushing buttons already everything is possible and available to them said m
r hanlon ultimately the consumer will tell the market they want of the new gadgets and
technologies being showcased at ces many of them are about enhancing the tv watching experience hi
gh_definition tv sets are everywhere and many new models of lcd liquid_crystal display tvs have be
en launched with dvr capability built into them instead of being external boxes one such example l
aunched at the show is humax inch lcd tv with an hour tivo dvr and dvd recorder one of the us bigg
est satellite tv companies directtv has even launched its own branded dvr at the show with hours o
f recording capability instant replay and search function the set can pause and rewind tv for up t
o hours and microsoft chief bill_gates announced in his pre show keynote_speech partnership with t
ivo called tivotogo which means people can play recorded programmes on windows pcs and mobile devi
ces all these reflect the increasing trend of freeing up multimedia so that people can watch what
they want when they want'
```

In [55]:

```
def make_bigrams(texts):
    return [bigram_mod[doc] for doc in texts]

def make_trigrams(texts):
    return [trigram_mod[bigram_mod[doc]] for doc in texts]
```

In [56]:

```
# Form Bigrams
data_words_bigrams = make_bigrams(data_words)
```

PCA plots to observe the behaviour of documents

In [57]:

```
from sklearn import model_selection, preprocessing, linear_model, naive_bayes, metrics
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer,HashingVectorizer
from sklearn import decomposition, ensemble
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
```

In [58]:

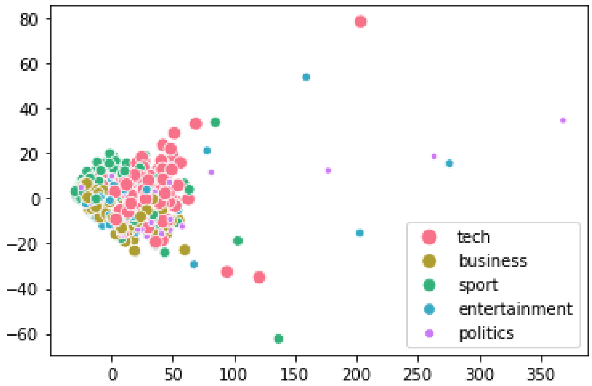
```
# Count Vectors as features
# create a count vectorizer object
count_vect = CountVectorizer(analyzer='word', token_pattern=r'\w{1,}')
count_vect.fit(unique_data_text)
```

Out[58]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='\\w{1,}', tokenizer=None,
                vocabulary=None)
```

In [59]:

```
xtrain_count = count_vect.transform(unique_data.text)

# plot the train features
pca = PCA(n_components=2).fit(xtrain_count.toarray())
data2D = pca.transform(xtrain_count.toarray())
cmap = sns.cubehelix_palette(dark=.3, light=.8, as_cmap=True)
ax = sns.scatterplot(data2D[:,0], data2D[:,1],
hue=unique_data.category.tolist(),size=unique_data.category.tolist(),palette="husl")
```



In [60]:

```
from sklearn.manifold import TSNE
```

In [61]:

```
def Pca_tf_idf_plot(text,label,min_word_len = 1,feature_size = 2000):
    tfidf_vect = TfidfVectorizer(analyzer='word', token_pattern=r'\w{'+ str(min_word_len) +',}', ma
x_features=feature_size)
    tfidf_vect.fit(text)
    text_tfidf =  tfidf_vect.transform(text)

    plt.figure(figsize=(15,10))

#     pca = PCA(n_components=2).fit(text_tfidf.toarray())
```

```
#     data2D = pca.transform(text_tfidf.toarray())
    data2D = TSNE(random_state=1).fit_transform(text_tfidf.toarray())
    cmap = sns.cubehelix_palette(dark=.3, light=.8, as_cmap=True)
    ax = sns.scatterplot(data2D[:,0], data2D[:,1], hue=label.tolist(),size=label.tolist(),palette="
husl")
```

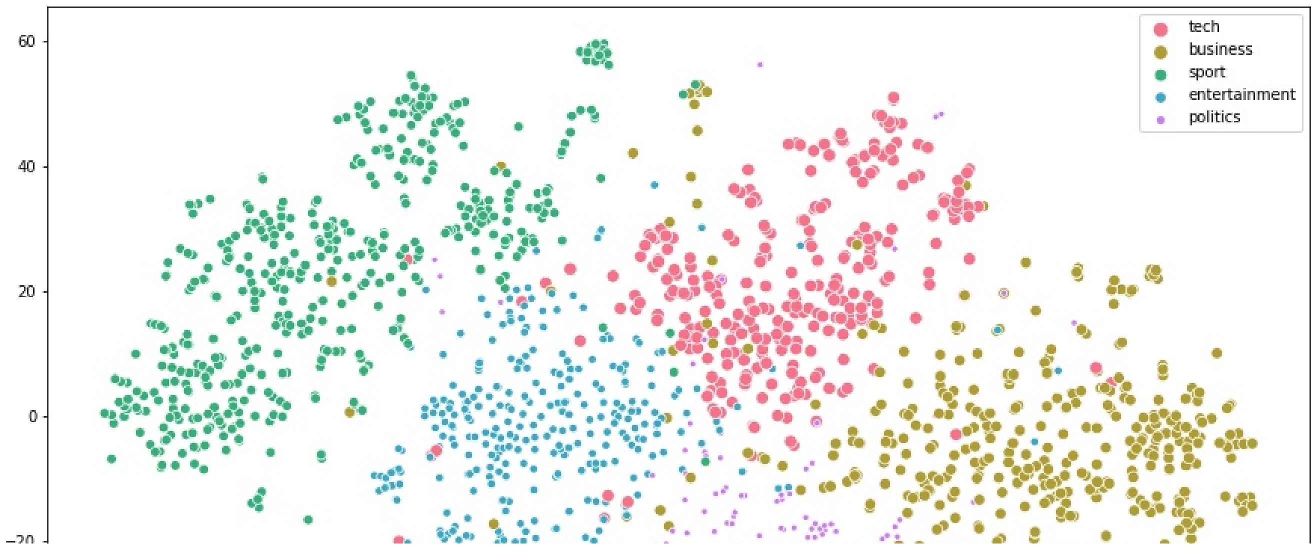
PCA plots of different feature_size and n_grams combination

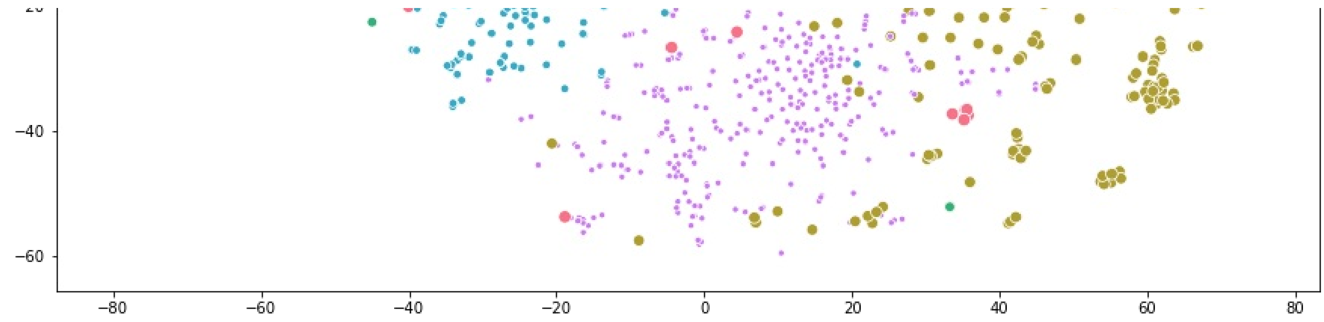
In [62]:

```
Pca_tf_idf_plot(text = unique_data.text, label = unique_data.category, min_word_len = 1, feature_si
ze=2000)
```

In [63]:

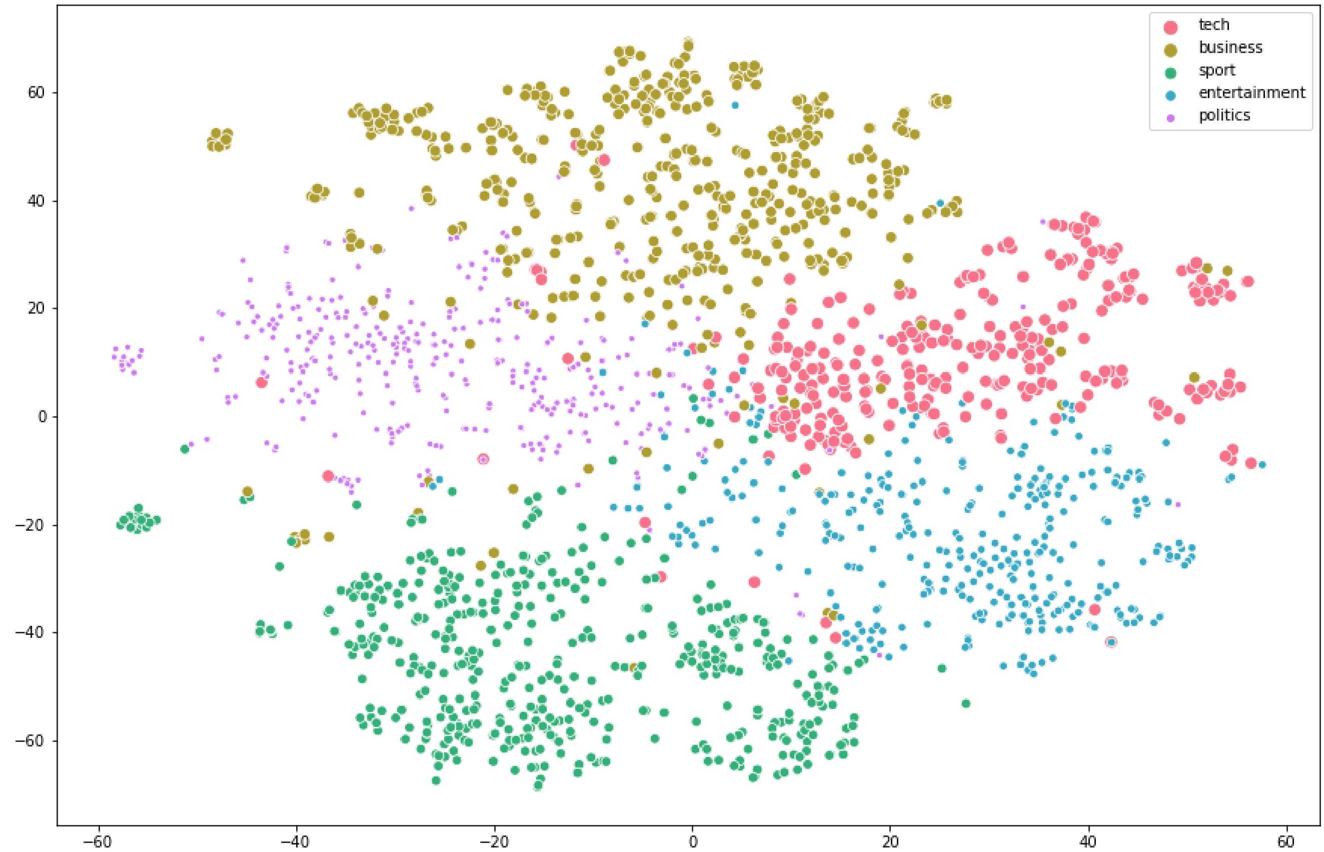
```
Pca_tf_idf_plot(text = unique_data.text, label = unique_data.category, min_word_len = 1, feature_si
ze=4000)
```





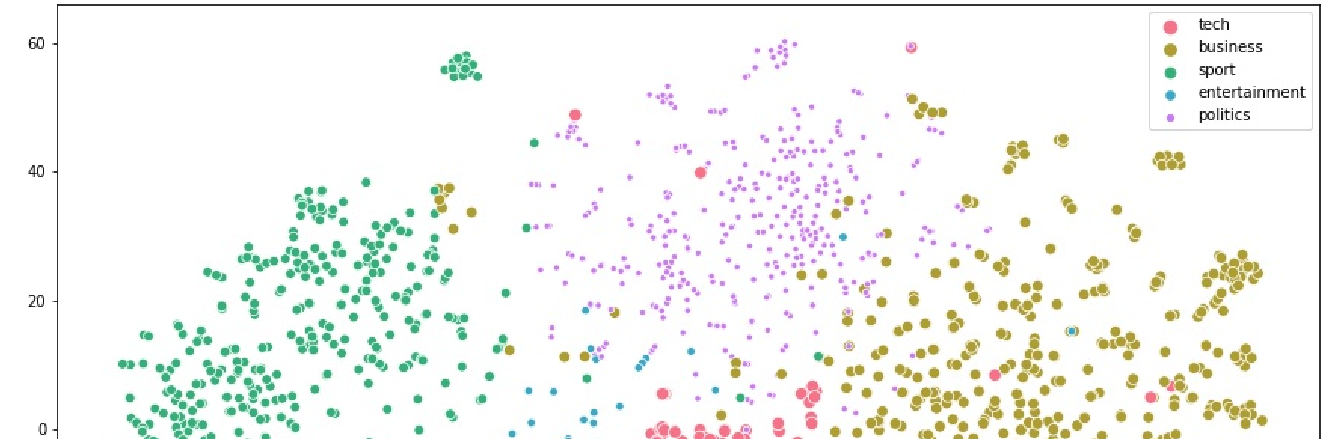
In [64]:

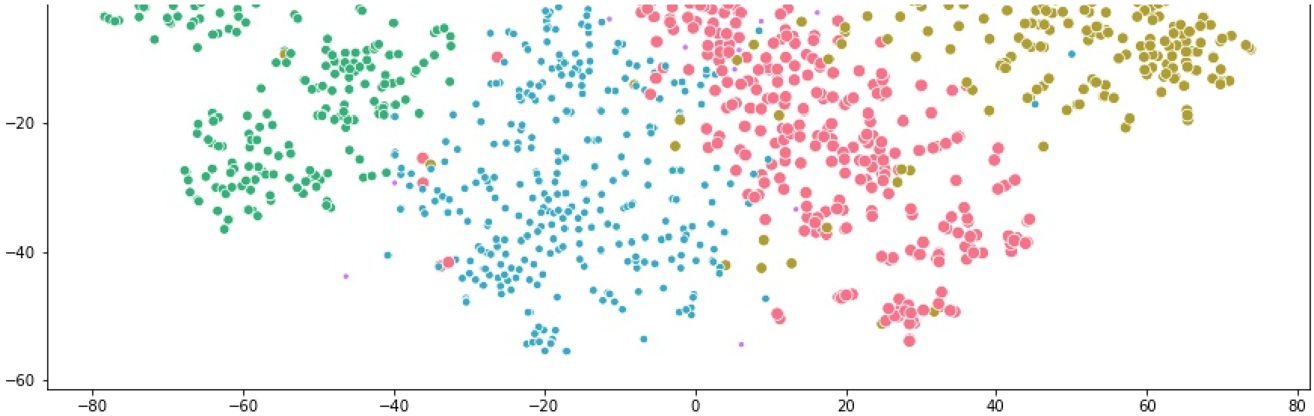
```
Pca_tf_idf_plot(text = unique_data.text, label = unique_data.category, min_word_len = 2, feature_si
ze=2000)
#plt.savefig("PCA_tfidf_vect_2_2000.png")
```



In [65]:

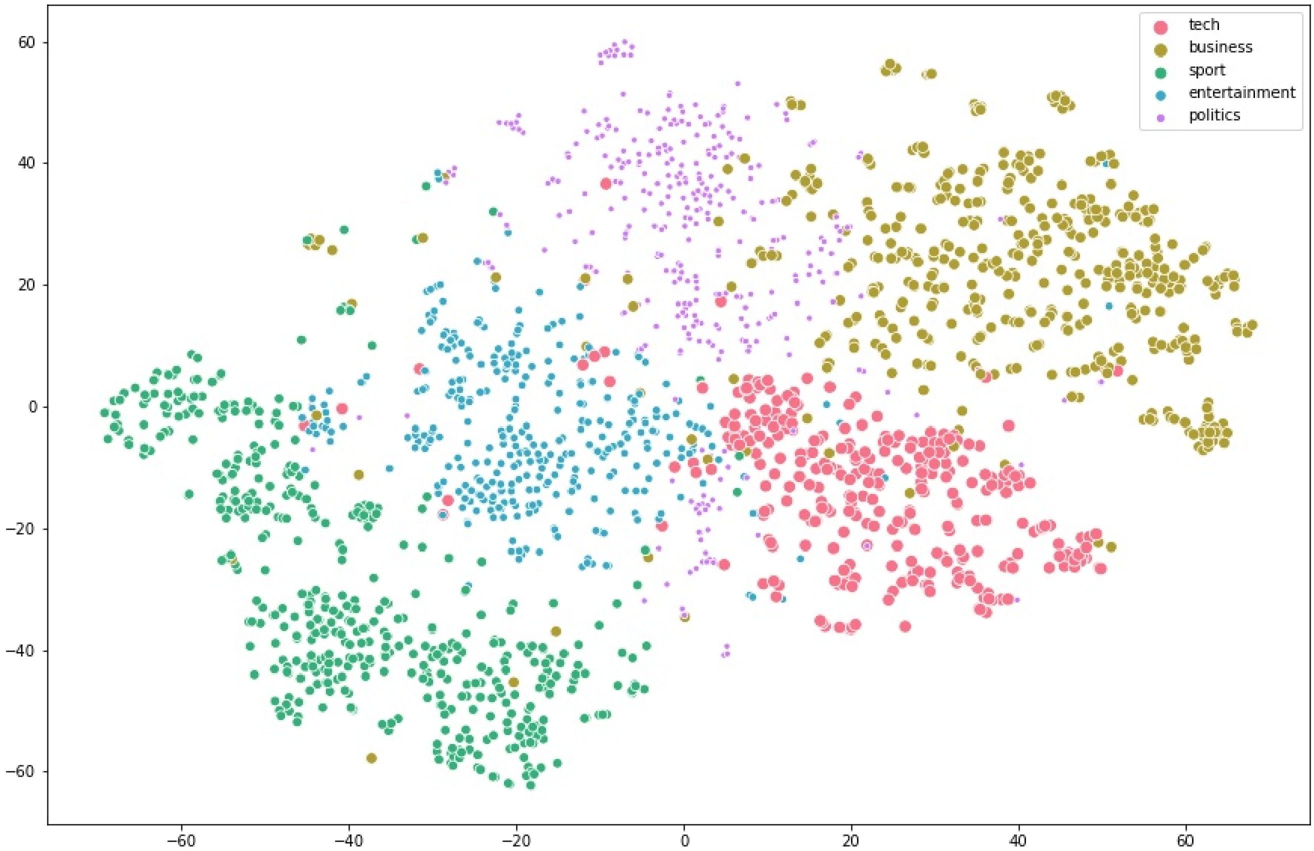
```
Pca_tf_idf_plot(text = unique_data.text, label = unique_data.category, min_word_len = 2, feature_si
ze=4000)
#plt.savefig("PCA_tfidf_vect_2_4000.png")
```





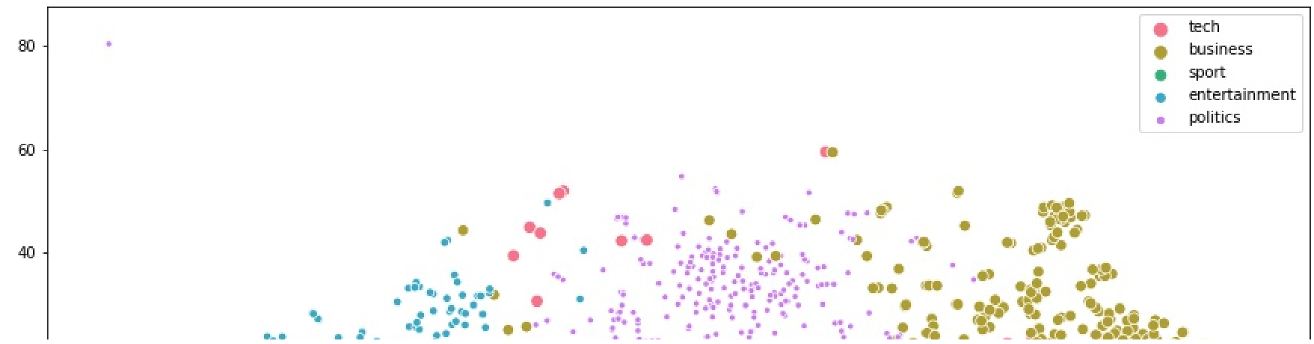
In [66]:

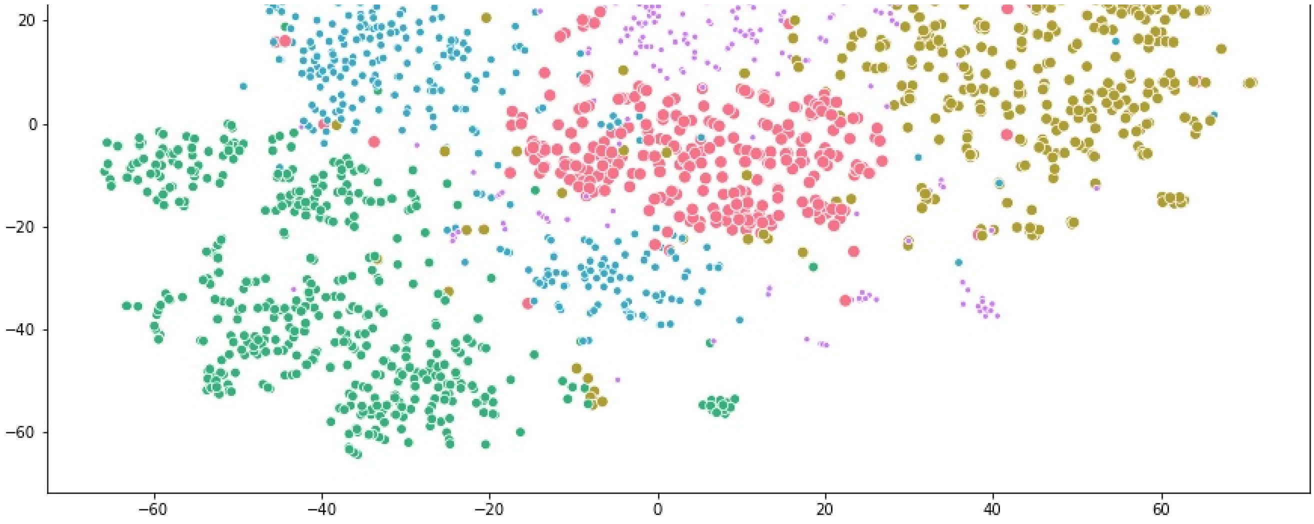
```
Pca_tf_idf_plot(text = unique_data.text, label = unique_data.category, min_word_len = 3, feature_si
ze=2000)
#plt.savefig("PCA_tfidf_vect_3_2000.png")
```



In [67]:

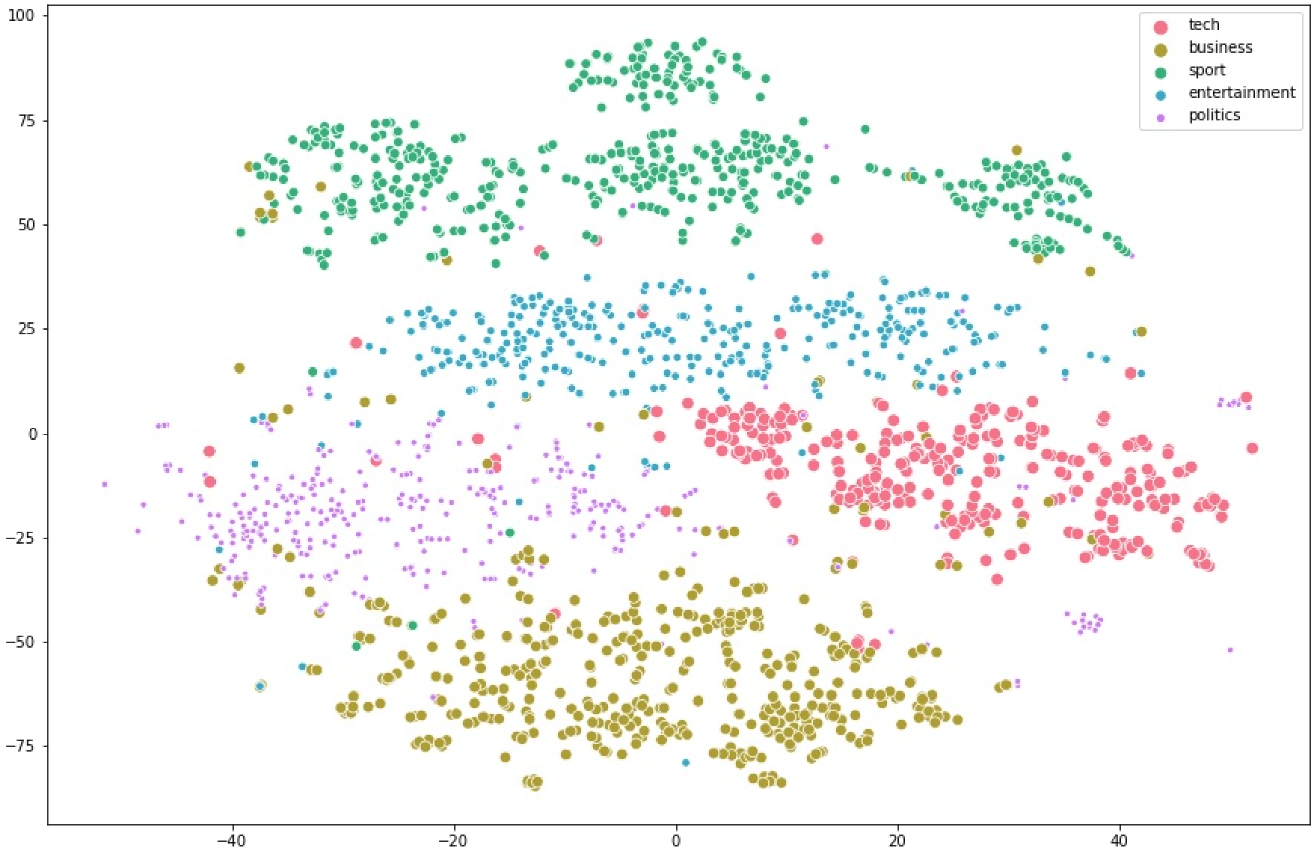
```
Pca_tf_idf_plot(text = unique_data.text, label = unique_data.category, min_word_len = 3, feature_si
ze=4000)
#plt.savefig("PCA_tfidf_vect_3_4000.png")
```





In [68]:

```
Pca_tf_idf_plot(text = unique_data.text, label = unique_data.category, min_word_len = 4, feature_si
ze=4000)
#plt.savefig("PCA_tfidf_vect_4_4000.png")
```

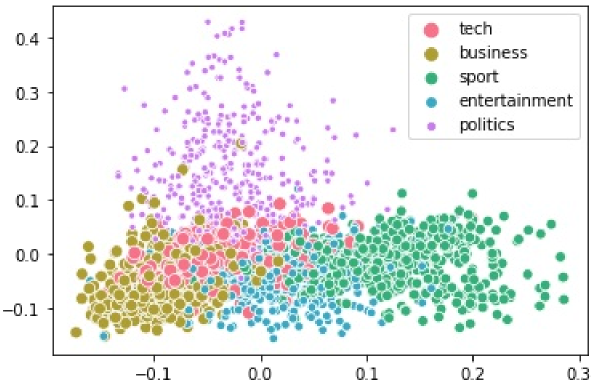


In [69]:

```
tfidf_vect = TfidfVectorizer(analyzer='word', token_pattern=r'\w{1,}', ngram_range=(2,3), max_featu
res=5000)
tfidf_vect.fit(unique_data.text)
text_tfidf =  tfidf_vect.transform(unique_data.text)
```

In [70]:

```
pca = PCA(n_components=2).fit(text_tfidf.toarray())
data2D = pca.transform(text_tfidf.toarray())
cmap = sns.cubehelix_palette(dark=.3, light=.8, as_cmap=True)
ax = sns.scatterplot(data2D[:,0], data2D[:,1],
hue=unique_data.category.tolist(),size=unique_data.category.tolist(),palette="husl")
```



Feature extraction from text using Tfidf

In [71]:

```
import re, string
import pandas as pd, numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
import os


vectorizer = TfidfVectorizer(ngram_range=(1,2),
               min_df=3, max_df=0.9, strip_accents='unicode', use_idf=1,
               smooth_idf=1, sublinear_tf=1, stop_words='english')
X = vectorizer.fit_transform(unique_data.text)
word_vects = X.toarray()
word_vects.shape
```

Out[71]:

```
(2126, 27662)
```

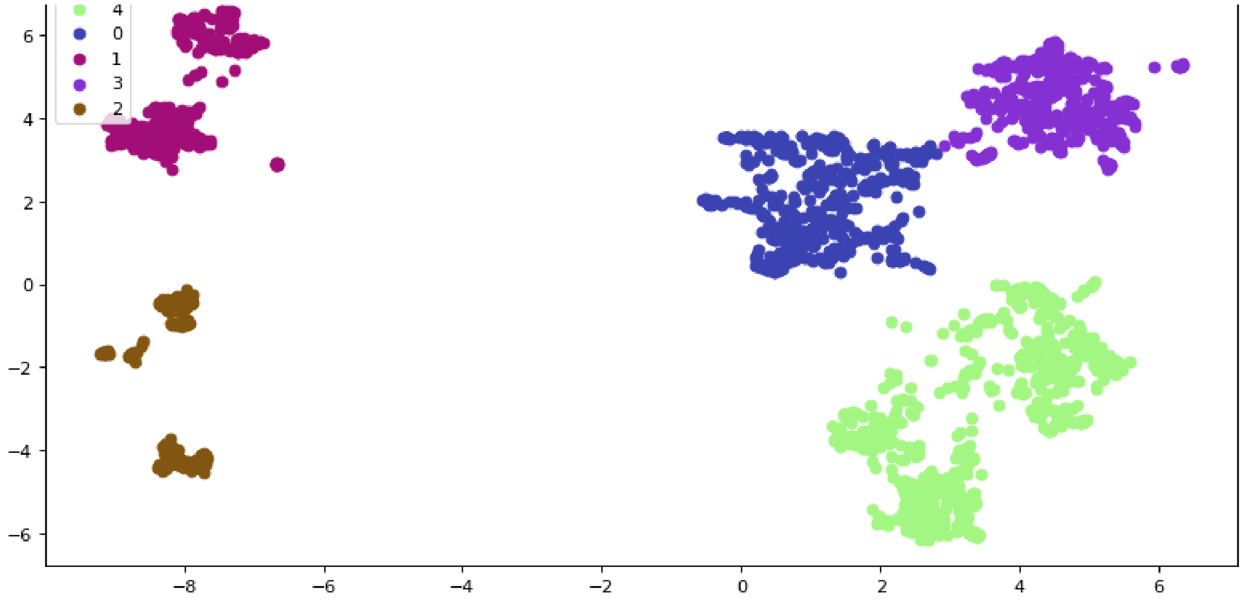
In [72]:

```
import umap

reducer = umap.UMAP(random_state=70,metric='cosine')
embedding = reducer.fit_transform(word_vects)
```

Plotting clusters to verify that our data can really separable or there are some noise and we should do more pre processing

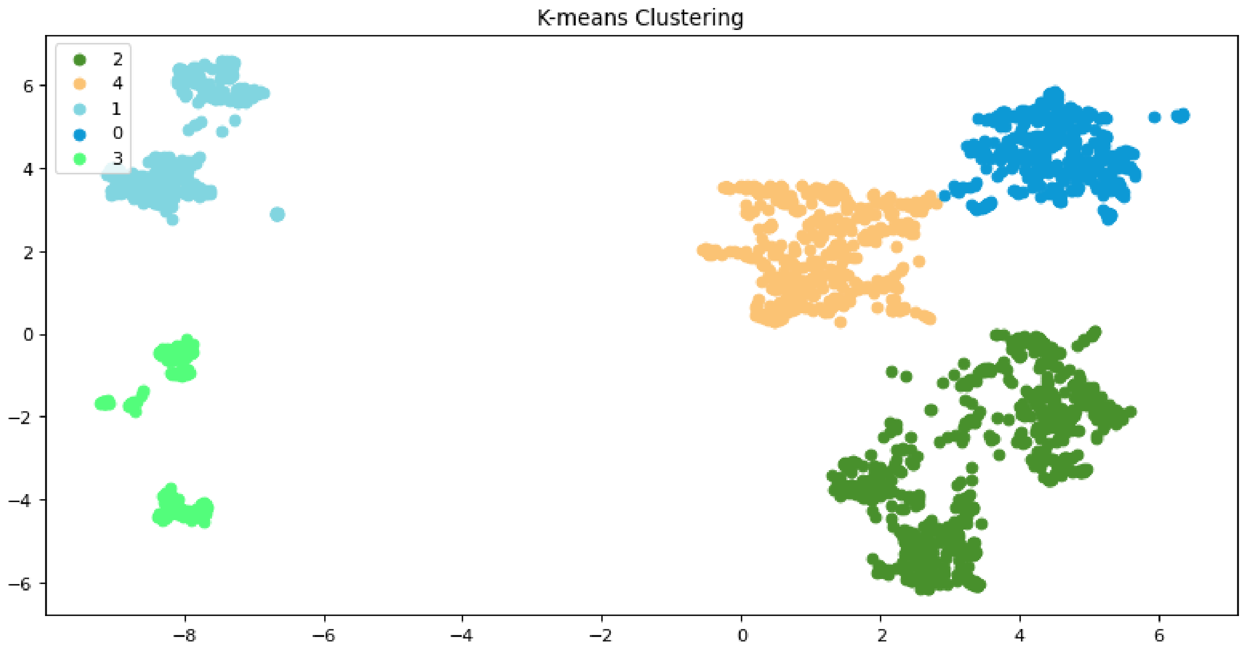
In [73]:

```
from sklearn.cluster import KMeans

clustering = KMeans(n_clusters=5, init='k-means++').fit(embedding)

unique_data['cluster'] = clustering.labels_
unique_data['vectX'] = embedding[:,0]
unique_data['vectY'] = embedding[:,1]
unique_data.cluster.unique()
plt.figure(num=None, figsize=(12, 6), dpi=80, facecolor='w', edgecolor='k')
for x in unique_data.cluster.unique():
    vctsX = unique_data.loc[unique_data.cluster == x].vectX
    vctsY = unique_data.loc[unique_data.cluster == x].vectY
    c = unique_data.loc[unique_data.cluster == x].cluster
    plt.title("K-means Clustering")
    plt.scatter(vctsX, vctsY, c=np.random.rand(3,), label=x)
    plt.legend(loc='upper left')
```



In [74]:

```
import umap

reducer = umap.UMAP(random_state=70,metric='cosine')
embedding = reducer.fit_transform(word_vects)

clustering = KMeans(n_clusters=5, init='k-means++').fit(embedding)

unique_data['cluster'] = clustering.labels_
unique_data['vectX'] = embedding[:,0]
unique_data['vectY'] = embedding[:,1]
unique_data.cluster.unique()
plt.figure(num=None, figsize=(12, 6), dpi=80, facecolor='w', edgecolor='k')
for x in unique_data.cluster.unique():
    vctsX = unique_data.loc[unique_data.cluster == x].vectX
    vctsY = unique_data.loc[unique_data.cluster == x].vectY
    c = unique_data.loc[unique_data.cluster == x].cluster
    plt.title("K-means Clustering")
    plt.scatter(vctsX, vctsY, c=np.random.rand(3,), label=x)
    plt.legend(loc='upper left')
```



We can see that our documents are clustered without much overlapping so now time to apply supervised ML methods to find the exact labels

In [75]:

```
tech = unique_data[unique_data.category == "tech"]
```

In [76]:

```
len(tech)
```

Out[76]:

```
347
```

Initial test on accuracy on different models

In [78]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB

from sklearn.model_selection import cross_val_score


models = [
    RandomForestClassifier(n_estimators=200, max_depth=10, random_state=0),
    MultinomialNB(),
    LogisticRegression(random_state=0),
]
```

In [79]:

```
CV = 5  # Cross Validate with 5 different folds of 20% data ( 80-20 split with 5 folds )

#Create a data frame that will store the results for all 5 trials of the 3 different models
cv_df = pd.DataFrame(index=range(CV * len(models)))
# Initially all entries are empty
```

In [80]:

```
unique_data
```

Out[80]:

	category	text	cluster	vectX	vectY
0	tech	tv future in the hands of viewers with home th...	2	4.524428	-2.257914
1	business	worldcom boss left books alone former worldc...	4	2.263807	0.563622
2	sport	tigers wary of farrell gamble leicester say ...	1	-7.184271	5.623127
3	sport	yeading face newcastle in fa cup premiership s...	1	-7.926629	3.391646
4	entertainment	ocean s twelve raids box office ocean s twelve...	2	3.375004	-6.022665
...	...	...	...	...	...
2220	business	cars pull down us retail figures us retail sal...	4	0.163443	2.935366
2221	politics	kilroy unveils immigration policy ex-chatshow ...	0	6.296341	5.243505
2222	entertainment	rem announce new glasgow concert us band rem h...	2	1.416165	-3.671859
2223	politics	how political squabbles snowball it s become c...	0	4.535325	5.649057
2224	sport	souness delight at euro progress boss graeme s...	1	-8.327559	4.125999

2126 rows × 5 columns

We will first convert the sentences into tfidf and then feed it into model

In [81]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5, norm='l2', encoding='latin-1', ngram_range=(1,
3), stop_words='english')

features = tfidf.fit_transform(unique_data.text).toarray() # Remaps the words in the 1490 articles
in the text column of
```

In [82]:

```
# Associate Category names with numerical index and save it in new column category_id
unique_data['category_id'] = unique_data['category'].factorize()[0]

#View first 10 entries of category_id, as a sanity check
unique_data['category_id'][0:10]
```

Out[82]:

```
0    0
1    1
2    2
3    2
4    3
5    4
6    4
7    2
8    2
9    3
Name: category_id, dtype: int64
```

In [83]:

```
datax = unique_data.text
```

In [84]:

```
labels = unique_data.category_id                          # represents the category of each of the
1490 articles
```

In [85]:

```
train_x = tfidf.fit_transform(datax[:1700]).toarray()
train_y = labels[:1700]
```

In [86]:

```
test_x = tfidf.transform(datax[1700:]).toarray()
test_y = labels[1700:]
```

In [87]:

```
# Create a new pandas dataframe "category_id_df", which only has unique Categories, also sorting t
his list in order of category_id values
category_id_df = unique_data[['category',
'category_id']].drop_duplicates().sort_values('category_id')
```

In [88]:

```
category_id_df
```

Out[88]:

	category	category_id
0	tech	0
1	business	1
2	sport	2

	category	category_id
2	sport	2
4	entertainment	3
5	politics	4

Creating ID and lables dictionary which is used to give label from ID after predictui

In [89]:

```
# Create a dictionary ( python datastructure - like a lookup table) that
# can easily convert category names into category_ids and vice-versa
category_to_id = dict(category_id_df.values)
id_to_category = dict(category_id_df[['category_id', 'category']].values)
```

In [90]:

```
# Create a dictionary ( python datastructure - like a lookup table) that
# can easily convert category names into category_ids and vice-versa
category_to_id = dict(category_id_df.values)
id_to_category = dict(category_id_df[['category_id', 'category']].values)
```

Applying Logistic model

In [91]:

```
# Pick 5 random samples from the dataframe
unique_data.sample(5, random_state=0)
```

Out[91]:

	category	text	cluster	vectX	vectY	category_id
664	entertainment	no uk premiere for rings musical the producers...	2	2.350062	-4.938293	3
1801	business	continental may run out of cash shares in co...	4	1.256449	0.539400	1
1258	business	honda wins china copyright ruling japan s hond...	4	2.470295	2.414011	1
1881	politics	woolf murder sentence rethink plans to give mu...	0	5.321232	4.086107	4
839	tech	format wars could confuse users technology f...	2	3.668677	-2.030583	0

In [92]:

```
logisticModel = LogisticRegression(random_state=0)
```

In [93]:

```
accuracies = cross_val_score(logisticModel, features, labels, scoring='accuracy', cv=CV)
```

In [94]:

```
logisticModel.fit(train_x, train_y)
```

Out[94]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Testing on single test document

In [95]:

```
datax[1701]
```

Out[95]:

```
'benitez  to launch morientes bid  liverpool may launch an £8m january bid for long-time target fe
rnando morientes  according to reports.  the real madrid striker has been linked with a move to an
field since the summer and is currently behind raul  ronaldo and michael owen at the bernabeu. liv
erpool boss rafael benitez is keen to bolster his forward options with djibril cisse out until
next season.  if there is an attractive propostition it could be i would be keen to leave   admitte
d the 28-year-old morientes. he added:  unfortunately  i m not in control of the situation. i m un
der contract to real and they will make any decisions.  the fee could put liverpool off a prospect
ive deal but real are keen to net the cash as they are reported to be preparing a massive summer b
id for inter milan striker adriano. the reds are currently sixth in the premiership  15 points beh
ind leaders chelsea.'
```

During test also we need to convert our sentence into tfidf for prediction

In [96]:

```
testSent = tfidf.transform([datax[1701]])
```

In [97]:

```
id_to_category[int(logisticModel.predict(testSent))]
```

Out[97]:

```
'sport'
```

In [98]:

```
# Function for prediction

def predictModel(text,tfidf_, model):
    testSent = tfidf_.transform([text])
    return id_to_category[int(logisticModel.predict(testSent))]
```

Fit on other models

On cross validation 5

In [99]:

```
entries = []
for model in models:
    model_name = model.__class__.__name__
  # create 5 models with different 20% test sets, and store their accuracies
    accuracies = cross_val_score(model, features, labels, scoring='accuracy', cv=CV)
  # Append all 5 accuracies into the entries list ( after all 3 models are run, there will be 3x5
= 15 entries)
    for fold_idx, accuracy in enumerate(accuracies):
        entries.append((model_name, fold_idx, accuracy))
```

In [100]:

```
# Store the entries into the results dataframe and name its columns
cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx', 'accuracy'])

# Mean accuracy of each algorithm
cv_df.groupby('model_name').accuracy.mean()
```

Out[100]:

```
model_name
LogisticRegression        0.980717
MultinomialNB             0.971307
RandomForestClassifier    0.928038
Name: accuracy, dtype: float64
```

In [101]:

```
cv_df
```

Out[101]:

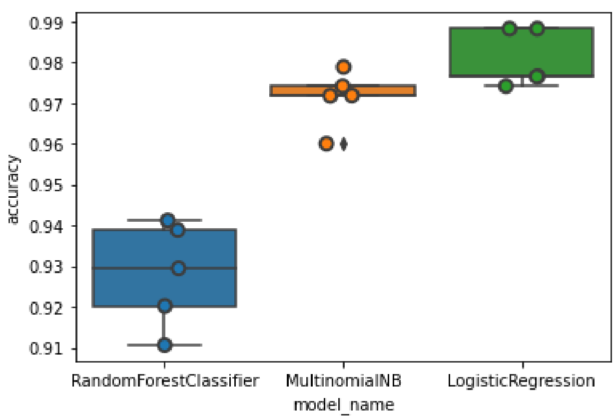
	model_name	fold_idx	accuracy
0	RandomForestClassifier	0	0.920188
1	RandomForestClassifier	1	0.941176
2	RandomForestClassifier	2	0.938824
3	RandomForestClassifier	3	0.910588
4	RandomForestClassifier	4	0.929412
5	MultinomialNB	0	0.971831
6	MultinomialNB	1	0.974118
7	MultinomialNB	2	0.978824
8	MultinomialNB	3	0.960000
9	MultinomialNB	4	0.971765
10	LogisticRegression	0	0.976526
11	LogisticRegression	1	0.976471
12	LogisticRegression	2	0.988235
13	LogisticRegression	3	0.974118
14	LogisticRegression	4	0.988235

In [102]:

```
import seaborn as sns

sns.boxplot(x='model_name', y='accuracy', data=cv_df)
sns.stripplot(x='model_name', y='accuracy', data=cv_df,
              size=8, jitter=True, edgecolor="gray", linewidth=2)
```

Out[102]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efdd493ed50>
```



On cross validation 10

In [103]:

```
entries = []
for model in models:
    model_name = model.__class__.__name__
  # create 5 models with different 20% test sets, and store their accuracies
    accuracies = cross_val_score(model, word_vects, labels, scoring='accuracy', cv=10)
  # Append all 5 accuracies into the entries list ( after all 3 models are run, there will be 3x5
  15 entries)
```

```
= 15 entries)
    for fold_idx, accuracy in enumerate(accuracies):
        entries.append((model_name, fold_idx, accuracy))
```

In [104]:

```
# Store the entries into the results dataframe and name its columns
cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx', 'accuracy'])

# Mean accuracy of each algorithm
cv_df.groupby('model_name').accuracy.mean()
```

Out[104]:

```
model_name
LogisticRegression        0.981185
MultinomialNB             0.969882
RandomForestClassifier    0.932244
Name: accuracy, dtype: float64
```

In [105]:

```
cv_df
```

Out[105]:

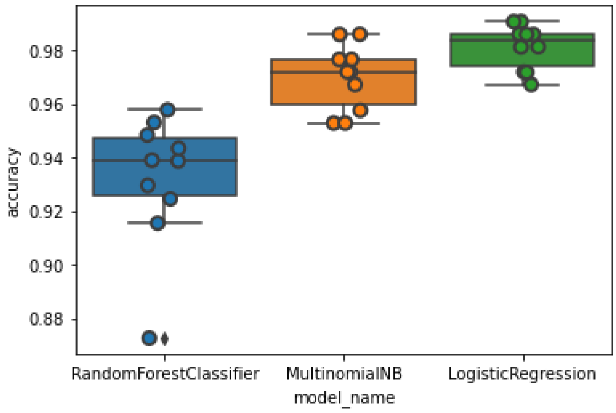
	model_name	fold_idx	accuracy
0	RandomForestClassifier	0	0.915493
1	RandomForestClassifier	1	0.929577
2	RandomForestClassifier	2	0.948357
3	RandomForestClassifier	3	0.938967
4	RandomForestClassifier	4	0.957746
5	RandomForestClassifier	5	0.953052
6	RandomForestClassifier	6	0.924528
7	RandomForestClassifier	7	0.872642
8	RandomForestClassifier	8	0.938679
9	RandomForestClassifier	9	0.943396
10	MultinomialNB	0	0.976526
11	MultinomialNB	1	0.971831
12	MultinomialNB	2	0.971831
13	MultinomialNB	3	0.967136
14	MultinomialNB	4	0.985915
15	MultinomialNB	5	0.976526
16	MultinomialNB	6	0.957547
17	MultinomialNB	7	0.952830
18	MultinomialNB	8	0.985849
19	MultinomialNB	9	0.952830
20	LogisticRegression	0	0.967136
21	LogisticRegression	1	0.990610
22	LogisticRegression	2	0.981221
23	LogisticRegression	3	0.971831
24	LogisticRegression	4	0.990610
25	LogisticRegression	5	0.985915
26	LogisticRegression	6	0.971698
27	LogisticRegression	7	0.981132
28	LogisticRegression	8	0.985849
29	LogisticRegression	9	0.985849

In [106]:

```
import seaborn as sns

sns.boxplot(x='model_name', y='accuracy', data=cv_df)
sns.stripplot(x='model_name', y='accuracy', data=cv_df,
              size=8, jitter=True, edgecolor="gray", linewidth=2)
```

Out[106]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efd9173b290>
```



In [107]:

```
from sklearn.model_selection import train_test_split


model = LogisticRegression(random_state=0)

#Split Data
X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features, labels,
unique_data.index, test_size=0.50, random_state=0)

#Train Algorithm
model.fit(X_train, y_train)

# Make Predictions
y_pred_proba = model.predict_proba(X_test)
y_pred = model.predict(X_test)
```

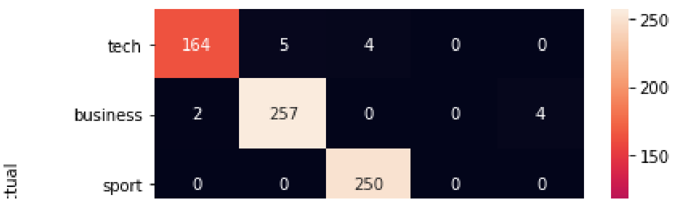
Confusion matrix for logistic

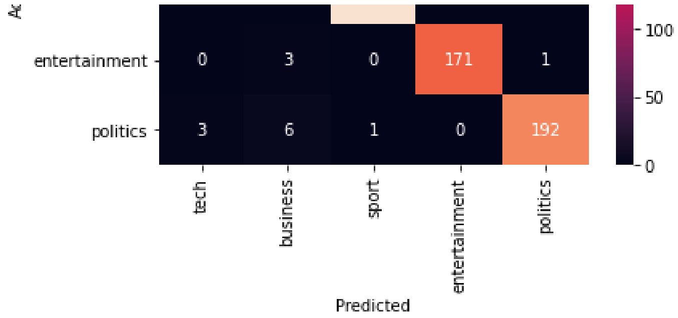
In [108]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

conf_mat = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_mat, annot=True, fmt='d',
            xticklabels=category_id_df.category.values, yticklabels=category_id_df.category.values)
plt.ylabel('Actual')
plt.xlabel('Predicted')
```

Out[108]:

```
Text(0.5, 15.0, 'Predicted')
```





Saving model for prediction directly along with tfidf transformer

In [109]:

```
import pickle
filename = 'finalized_model.model'
pickle.dump(logisticModel, open(filename, 'wb'))
# Dump the file
pickle.dump(tfidf, open("tfidftransformer.tfidf", "wb"))
```

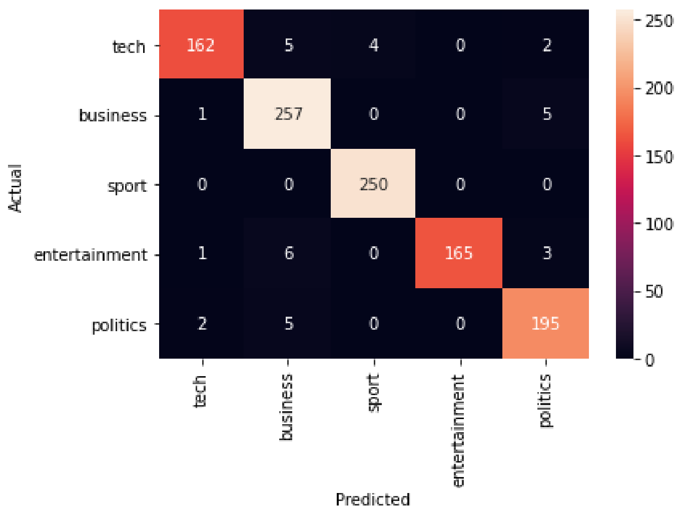
Prediction on Multonomial naive bayes and its confusion matrix

In [110]:

```
model = MultinomialNB()

#Split Data
X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features, labels,
unique_data.index, test_size=0.50, random_state=0)

#Train Algorithm
model.fit(X_train, y_train)

# Make Predictions
y_pred_proba = model.predict_proba(X_test)
y_pred = model.predict(X_test)


conf_mat = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_mat, annot=True, fmt='d',
            xticklabels=category_id_df.category.values, yticklabels=category_id_df.category.values)
plt.ylabel('Actual')
plt.xlabel('Predicted')
#plt.savefig("Naive_bayes_matrix_logistic.png")
```

Out[110]:

```
Text(0.5, 15.0, 'Predicted')
```



classification using NER

In [111]:

```
import spacy
from spacy import displacy
from collections import Counter
import en_core_web_sm
nlp = en_core_web_sm.load()
```

In [112]:

```
s = 'European authorities fined Google a record $5.1 billion on Wednesday for abusing its power in
the mobile phone market and ordered the company to alter its practices'
doc = nlp(s)
print([(X.text, X.label_) for X in doc.ents])
```

```
[('European', 'NORP'), ('Google', 'ORG'), ('$5.1 billion', 'MONEY'), ('Wednesday', 'DATE')]
```

In [113]:

```
newString = s
for e in reversed(doc.ents): #reversed to not modify the offsets of other entities when
substituting
    start = e.start_char
    end = start + len(e.text)
    newString = newString[:start] + e.label_ + newString[end:]
print(newString)
```

```
NORP authorities fined ORG a record MONEY on DATE for abusing its power in the mobile phone market
and ordered the company to alter its practices
```

Replacing name entity with its class using NER

In [114]:

```
def replace_token_with_entity(s):
    doc = nlp(s)
    newString = s
    for e in reversed(doc.ents): #reversed to not modify the offsets of other entities when
substituting
        start = e.start_char
        end = start + len(e.text)
        newString = newString[:start] + e.label_ + newString[end:]
    return newString
```

In [115]:

```
replace_token_with_entity(s)
```

Out[115]:

```
'NORP authorities fined ORG a record MONEY on DATE for abusing its power in the mobile phone marke
t and ordered the company to alter its practices'
```

In [116]:

```
# ner_text = replace_token_with_entity(s)
unique_data
```

Out[116]:

	category	text	cluster	vectX	vectY	category_id
0	tech	tv future in the hands of viewers with home th...	2	4.524428	-2.257914	0

	category	text	cluster	vectX	vectY	category_id
1	business	worldcom boss left books alone former worldc...	4	2.263807	0.563622	1
2	sport	tigers wary of farrell gamble leicester say ...	1	-7.184271	5.623127	2
3	sport	yeading face newcastle in fa cup premiership s...	1	-7.926629	3.391646	2
4	entertainment	ocean s twelve raids box office ocean s twelve...	2	3.375004	-6.022665	3
...	...	...	...	...	...	...
2220	business	cars pull down us retail figures us retail sal...	4	0.163443	2.935366	1
2221	politics	kilroy unveils immigration policy ex-chatshow ...	0	6.296341	5.243505	4
2222	entertainment	rem announce new glasgow concert us band rem h...	2	1.416165	-3.671859	3
2223	politics	how political squabbles snowball it s become c...	0	4.535325	5.649057	4
2224	sport	souness delight at euro progress boss graeme s...	1	-8.327559	4.125999	2

2126 rows × 6 columns

In [117]:

```
unique_data.head()
```

Out[117]:

	category	text	cluster	vectX	vectY	category_id
0	tech	tv future in the hands of viewers with home th...	2	4.524428	-2.257914	0
1	business	worldcom boss left books alone former worldc...	4	2.263807	0.563622	1
2	sport	tigers wary of farrell gamble leicester say ...	1	-7.184271	5.623127	2
3	sport	yeading face newcastle in fa cup premiership s...	1	-7.926629	3.391646	2
4	entertainment	ocean s twelve raids box office ocean s twelve...	2	3.375004	-6.022665	3

In [118]:

```
unique_data.text.iloc[1]
```

Out[118]:

```
'worldcom boss  left books alone  former worldcom boss bernie ebbers  who is accused of overseeing
an $11bn (£5.8bn) fraud  never made accounting decisions  a witness has told jurors.  david myers
made the comments under questioning by defence lawyers who have been arguing that mr ebbers was no
t responsible for worldcom s problems. the phone company collapsed in 2002 and prosecutors claim t
hat losses were hidden to protect the firm s shares. mr myers has already pleaded guilty to fraud
and is assisting prosecutors.  on monday  defence lawyer reid weingarten tried to distance his
client from the allegations. during cross examination  he asked mr myers if he ever knew mr ebbers
make an accounting decision  .  not that i am aware of   mr myers replied.  did you ever know mr e
bbers to make an accounting entry into worldcom books   mr weingarten pressed.  no   replied the wi
tness. mr myers has admitted that he ordered false accounting entries at the request of former
worldcom chief financial officer scott sullivan. defence lawyers have been trying to paint mr
sullivan  who has admitted fraud and will testify later in the trial  as the mastermind behind wor
ldcom s accounting house of cards.  mr ebbers  team  meanwhile  are looking to portray him as an
affable boss  who by his own admission is more pe graduate than economist. whatever his abilities
mr ebbers transformed worldcom from a relative unknown into a $160bn telecoms giant and investor d
arling of the late 1990s. worldcom s problems mounted  however  as competition increased and the t
elecoms boom petered out. when the firm finally collapsed  shareholders lost about $180bn and 20 0
00 workers lost their jobs. mr ebbers  trial is expected to last two months and if found guilty th
e former ceo faces a substantial jail sentence. he has firmly declared his innocence.'
```

In [119]:

```
unique_data['ner_text'] = ""
for j in range(len(unique_data)):
    s = unique_data.text.iloc[j]
    ner_text = replace_token_with_entity(s)
    unique_data['ner_text'].iloc[j] = ner_text
```

```
/opt/conda/lib/python3.7/site-packages/pandas/core/indexing.py:671: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_with_indexer(indexer, value)
```

In [120]:

```
unique_data.ner_text.iloc[1]
```

Out[120]:

```
'ORG boss  left books alone  former ORG boss bernie ebbers  who is accused of overseeing an $MONEY (£MONEY) fraud  never made accounting decisions  a witness has told jurors.  PERSON made the comments under questioning by defence lawyers who have been arguing that mr ebbers was not responsible for ORG PRODUCT problems. the phone company collapsed in DATE and prosecutors claim that losses were hidden to protect the firm s shares. mr PERSON has already pleaded guilty to fraud and is assisting prosecutors.  on DATE  defence lawyer reid weingarten tried to distance his client from the allegations. during cross examination  he asked mr PERSON if he ever knew mr ebbers  make an accounting decision  .  not that i am aware of   mr PERSON replied.  did you ever know mr ebbers to make an accounting entry into ORG books   mr PERSON pressed.  no   replied the witness. mr PERSON has admitted that he ordered false accounting entries at the request of former ORG chief financial officer PERSON. defence lawyers have been trying to paint mr PERSON  who has admitted fraud and will testify later in the trial  as the mastermind behind ORG PRODUCT accounting house of cards.  mr ebbers  team  meanwhile  are looking to portray him as an affable boss  who by his own admission is more pe graduate than economist. whatever his abilities  mr ebbers transformed ORG from a relative unknown into a $MONEY telecoms giant and investor darling of DATE. ORG s problems mounted  however  as competition increased and the telecoms boom petered out. when the firm finally collapsed  shareholders lost MONEY and CARDINAL workers lost their jobs. mr ebbers  trial is expected to DATE and if found guilty the former ceo faces a substantial jail sentence. he has firmly declared his innocence.'
```

Fitting different models using NER generated text

In [121]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5, norm='l2', encoding='latin-1', ngram_range=(1, 2), stop_words='english')

features = tfidf.fit_transform(unique_data.ner_text).toarray() # Remaps the words in the 1490 articles in the text column of

entries = []
for model in models:
    model_name = model.__class__.__name__
  # create 5 models with different 20% test sets, and store their accuracies
    accuracies = cross_val_score(model, features, labels, scoring='accuracy', cv=CV)
  # Append all 5 accuracies into the entries list ( after all 3 models are run, there will be 3x5 = 15 entries)
    for fold_idx, accuracy in enumerate(accuracies):
        entries.append((model_name, fold_idx, accuracy))


# Store the entries into the results dataframe and name its columns
cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx', 'accuracy'])

# Mean accuracy of each algorithm
cv_df.groupby('model_name').accuracy.mean()
```

Out[121]:

```
model_name
LogisticRegression        0.971780
MultinomialNB             0.965663
RandomForestClassifier    0.925691
Name: accuracy, dtype: float64
```

In [122]:

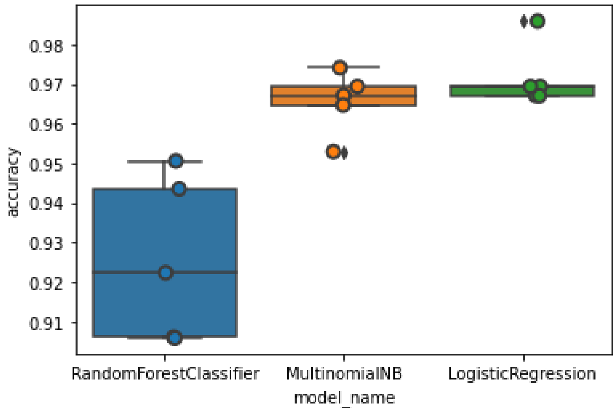
```
cv_df
```

Out[122]:

	model_name	fold_idx	accuracy
0	RandomForestClassifier	0	0.906103
1	RandomForestClassifier	1	0.943529
2	RandomForestClassifier	2	0.950588
3	RandomForestClassifier	3	0.905882
4	RandomForestClassifier	4	0.922353
5	MultinomialNB	0	0.967136
6	MultinomialNB	1	0.964706
7	MultinomialNB	2	0.974118
8	MultinomialNB	3	0.952941
9	MultinomialNB	4	0.969412
10	LogisticRegression	0	0.967136
11	LogisticRegression	1	0.969412
12	LogisticRegression	2	0.985882
13	LogisticRegression	3	0.967059
14	LogisticRegression	4	0.969412

In [123]:

```
import seaborn as sns

sns.boxplot(x='model_name', y='accuracy', data=cv_df)
sns.stripplot(x='model_name', y='accuracy', data=cv_df,
              size=8, jitter=True, edgecolor="gray", linewidth=2)
plt.savefig('NER_accuracy_boxplot.png')
```



Applying LDA(Latent Dirichlet Allocation) for Topic Modelling

In [124]:

```
#  Initialise the count vectorizer with the English stop words
count_vectorizer = CountVectorizer(stop_words='english')
# Fit and transform the processed titles
count_data = count_vectorizer.fit_transform(unique_data['text'])
```

In [125]:

```
import warnings
warnings.simplefilter("ignore", DeprecationWarning)
# Load the LDA model from sk-learn
from sklearn.decomposition import LatentDirichletAllocation as LDA

# Helper function
def print_topics(model, count_vectorizer, n_top_words):
```

```
    words = count_vectorizer.get_feature_names()
    for topic_idx, topic in enumerate(model.components_):
        print("\nTopic #%d:" % topic_idx)
        print(" ".join([words[i]
                        for i in topic.argsort()[:-n_top_words - 1:-1]]))

# Tweak the two parameters below
number_topics = 5
number_words = 15
# Create and fit the LDA model
lda = LDA(n_components=number_topics, n_jobs=-1)
lda.fit(count_data)
# Print the topics found by the LDA model
print("Topics found via LDA:")
print_topics(lda, count_vectorizer, number_words)
```

```
Topics found via LDA:

Topic #0:
said england year win game world play time new cup second match set france open

Topic #1:
film best said music year awards award new won mr director star band years actor

Topic #2:
said people mr new government technology use mobile make says uk service games like time

Topic #3:
said year market new growth sales 2004 time economy china firm bank company game prices

Topic #4:
mr said labour election party blair government brown people minister new tax year howard world
```

In [126]:

```
#  Initialise the count vectorizer with the English stop words
count_vectorizer = CountVectorizer(stop_words='english')
# Fit and transform the processed titles
count_data = count_vectorizer.fit_transform(unique_data[unique_data.category == "politics"].text)

# Tweak the two parameters below
number_topics = 3
number_words = 15
# Create and fit the LDA model
lda = LDA(n_components=number_topics, n_jobs=-1)
lda.fit(count_data)
# Print the topics found by the LDA model
print("Topics found via LDA:")
print_topics(lda, count_vectorizer, number_words)
```

```
Topics found via LDA:

Topic #0:
said mr party people new ukip kilroy silk government police year law hunting told plans

Topic #1:
said mr government home uk rights lord secretary people law human lords blunkett told house

Topic #2:
mr said labour election blair government party brown people minister howard prime tax tory
chancellor
```